Indice

Introduzione

Sono molti i livelli e gli ambiti applicativi relativamente ai quali è possibile trattare l'argomento posto a titolo. E' bene fornire qualche precisazione in merito, in modo da contestualizzare le presenti pagine.

Il livello di trattazione è quello utile al progettista di piccoli sistemi a microcontrollore. Particolare attenzione è quindi posta agli aspetti hardware della progettazione, benché vengano presentate anche alcune considerazioni di carattere software.

Gli ambiti applicativi sono quelli tipicamente industriali, nei quali la comunicazione seriale si sviluppa tipicamente su corte distanze (qualche centinaio di metri) e a bassa velocità (qualche centinaio di kbps).

I presenti appunti sono frutto, in gran parte, di un'accurata ricerca puntualmente riportata in bibliografia. Si consiglia la lettura del materiale riportato in caso di necessità di approfondimento.

Eventuali commenti e/o segnalazioni d'errore possono essere notificati a

`bandiziol@katamail.com`

Grazie.

San Giovanni al Natisone, 04/02/2015

Santino Bandiziol

Convenzioni adottate nel testo

Il testo normale è scritto utilizzando il presente carattere tipografico e stile di scrittura. Si è cercato di evitare inutili inglesismi ed un eccesso di acronimi, ma trattandosi di appunti tecnici è inevitabile il ricorso all'inglese tecnico e a frequenti abbreviazioni.

Nella prima eventualità, il termine in inglese incontrato per la prima volta viene scritto in corsivo, come nel caso della parola *font* (carattere tipografico). Nell'ipotesi in cui sia ritenuto utile, fra parentesi viene indicata una possibile traduzione che tenga conto del contesto.

Quando lo si ritiene utile la presente convenzione viene usata a "rovescio", ponendo tra parentesi la traduzione inglese di un termine italiano, come nel caso in cui si voglia parlare del carattere tipografico (*font*) e fornirne la traduzione.

L'uso degli acronimi segue regole simili. Viene indicato l'acronimo in grassetto e tra parentesi il suo significato, come nel caso dell'acronimo **PC** (Personal Computer). A volte la parentesi è omessa, come nel caso in cui la spegazione del significato venga fornita in forma discorsiva e non didascalica. Successivi usi dello stesso termine non prevedono alcuna ulteriore indicazione del significato posto fra parentesi. Siccome ciò può portare a qualche problema di lettura, il lettore può far riferimento, se presente, al glossario.

Il corsivo viene utilizzato anche per termini e frasi in italiano che si intendono *enfatizzare*, come nel presente caso. Si è comunque cercato di non abusare di tale convenzione.

L'icona di pericolo è utilizzata per richiamare l'attenzione del lettore su un passaggio particolarmente importante. E' bene, quindi, leggere con attenzione quanto evidenziato dalla presenza del triangolo di pericolo.

Capitolo 1

Principi fondamentali

Trattare il tema della comunicazione fra dispositivi elettronici è oggi ben diverso da quel che era qualche decennio fa. Il lettore è solitamente "polarizzato" e pensa immediatamente ai PC, ai telefonini, ai tablet, ecc., trascurando una nicchia piuttosto importante delle comunicazioni seriali che è quella delle comunicazioni fra dispositivi di controllo dei processi industriali, ovvero delle reti industriali.

Vale la pena soffermarsi su questi ultimi due termini.

Il termine "rete" è diventato di uso comune al punto tale che anche il non professionista è in grado di elencare vari tipi di reti: la rete telefonica, la rete Internet, le trasmissioni TV, ecc. Si tratta quindi, intuitivamente, di una serie di connessioni, più o meno estesa, atta a collegare diverse entità finalizzate o meno ad un obiettivo particolare.

La rete telefonica connette diversi utenti dando loro la possibilità di comunicare fra loro. Le trasmissioni TV inviano lo stesso segnale a diversi utenti che ne possono usufruire senza modificarlo. La rete Internet connette diversi PC fra loro per i fini più disparati.

Il termine "industriale" indica, invece, una fondamentale differenza con tutte le reti precedentemente citate: *le comunicazioni fra un elemento e l'altro della rete hanno un alto grado di affidabilità e avvengono sempre in un tempo assolutamente certo*. Quest'ultimo particolare, che sembra del tutto marginale, è invece molto importante, al punto tale da rappresentare un discrimine ineludibile nel progetto di una rete industriale.

Si supponga, ad esempio, il tentativo di autenticazione di un utente mediante un PC posto all'interno di una piccola rete aziendale o scolastica. L'utente digita il proprio *username* e la propria *password* sulla tastiera e invia la richiesta di autenticazione al *Server* di rete per la verifica.

Questa azione, così semplice e apparentemente poco problematica, potrebbe incontrare diversi ostacoli prima di essere esaudita: il messaggio inoltrato al Server potrebbe collidere con altri messaggi inviati "contemporaneamente" in rete da altri utenti; il Server potrebbe essere impegnato nel riordino di

una *directory* particolarmente voluminosa e non asservire immediatamente la richiesta; lo *username* potrebbe essere stato digitato in maniera non corretta in seguito ad un errore umano; la *password* potrebbe essere scaduta e l'utente potrebbe essere invitato a digitarne una nuova, ecc.

Tutti questi momentanei impedimenti, pur non compromettendo definitivamente l'accesso alla rete, producono il risultato di un differimento temporaneo nell'autenticazione dell'utente. Da pochi millisecondi, se tutto va bene, a qualche secondo.

Ciò può rappresentare, nel peggiore dei casi, una piccola noia per l'utente, ma non compromettere il buon funzionamento della rete o addirittura costituire motivo di abbandono di tale tecnologia.

Se i suddetti ritardi avvengono, invece, all'interno di una rete industriale, i danni potrebbero essere ingentissimi. Pochi secondi di ritardo nell'attivazione di un allarme durante una colata continua in un'acciaieria e vi potrebbe essere fuoriuscita di acciaio fuso dalla billetta all'uscita dalla lingottiera, con danni ingentissimi all'impianto o addirittura pericolo di vita per gli addetti alla colata.

Ciò perché nel caso nella rete industriale i tempi sono commisurati alle esigenze del controllo di processo e non a quelle, più elastiche, della pazienza umana.

Durante tutto il presente corso si dovrà tenere presente questa piccola ma fondamentale differenza fra rete industriale e rete di *office automation*. Alla luce di ciò verranno illustrate tecnologie molto differenti fra loro e si cercherà di spiegare la difficoltà che ha attualmente la tecnologia Ethernet ad entrare attivamente nell'industria e quali sono le risposte alle esigenze appena descritte.

Prima, però, si dovranno rinfrescare alcuni principi fondamentali che sono alla base delle comunicazioni fra PC al fine di gettare le basi di una terminologia comune e di un comune *background* tecnico.

1.1 La comunicazione

All'interno di un qualsiasi sistema complesso la comunicazione fra gli elementi che lo compongono assume un'importanza fondamentale. Ad essa viene demandato il delicato compito di trasferire le informazioni relative al mondo circostante, spesso ostile e non strutturato per agevolare il flusso comunicativo, da un elemento all'altro.

Questo vale sia in presenza di un sistema sociale, dove i singoli elementi sono rappresentati da esseri umani o animali, oppure di un sistema automatico, dove i singoli elementi costituenti la rete di comunicazione sono, solitamente, dei dispositivi elettronici.

Dalla bontà della comunicazione dipende spesso la bontà del sistema o, quanto meno, si può certamente asserire che la qualità della comunicazione è almeno uno dei parametri che caratterizzano la qualità del sistema. Ad influenzare l'efficacia della comunicazione, sia nei sistemi automatici che i quelli umani, concorrono molti elementi, tutti importanti:

- la sorgente, ossia l'elemento che dà inizio alla comunicazione;
- il destinatario, ossia l'elemento, intermedio o meno, a cui è destinata la comunicazione;
- l'oggetto della comunicazione;
- la forma di comunicazione usata;
- il mezzo utilizzato per effettuare la comunicazione;
- l'ambiente circostante entro il quale la comunicazione avviene.

E' evidente che la sorgente influenza in modo determinante la comunicazione. Se chi comunica non lo fa in maniera efficace, il messaggio non verrà mai interpretato correttamente.

Pure un cattivo ascoltatore pregiudica l'efficacia della comunicazione. Il destinatario del messaggio deve essere in grado di comprenderlo, qualora esso sia comunicato in modo corretto e senza interferenze.

Anche l'oggetto della comunicazione concorre a determinare il suo successo. Si pensi a quanto possa a volte essere difficile, ad esempio, comunicare a qualcuno un'emozione. A volte ciò risulta difficile anche fra persone con ottima proprietà di linguaggio. In un sistema automatico possono nascere problemi se l'oggetto della trasmissione necessita di un tempo molto lungo.

La forma utilizzata è altrettanto importante. Si pensi alla comunicazione nel mondo animale o ai linguaggi non verbali.

Essa non va confusa con il mezzo utilizzato per veicolare l'informazione, ovvero come essa venga "portata" da un elemento all'altro.

Infine, un ambiente ostile può ostacolare una efficace comunicazione, intervenendo negativamente sulla sorgente, sul destinatario oppure sul mezzo di comunicazione.

Ciascuno dei suddetti elementi verrà, direttamente o indirettamente, analizzato nel corso del presente documento, anche se non nell'ordine appena presentato.

Converrà iniziare l'analisi partendo dalle forme utilizzate per comunicare, concentrandoci su una in particolare. Le forme di comunicazione, infatti, sono moltissime, ma fra quelle più evolute si possono certamente collocare quelle che utilizzano, in vario modo, i linguaggi.

Fra questi ultimi, infine, di particolare interesse sono i *linguaggi formali*, di cui brevemente ci occuperemo.

1.1.1 Il linguaggio

Dire che il linguaggio è la forma più evoluta di comunicazione, oltre a essere un'affermazione probabilmente faziosa, visto che è fatta utilizzando un linguaggio, è anche ambigua e imprecisa, dato che si omette di indicare il tipo di linguaggio al quale ci si riferisce.

C'è molta differenza, infatti, fra alcuni linguaggi non verbali (si pensi al corteggiamento fra animali, ad esempio) e i linguaggi formali. Questi ultimi tendono a minimizzare i malintesi fra sorgente e destinatario, ovvero ad occuparsi dell'integrità e univocità interpretativa del messaggio. Ciò avviene rendendo chiari ed espliciti gli elementi costituenti il linguaggio.

Ogni linguaggio si basa su un *vocabolario*, i cui elementi sono solitamente chiamati parole, ma, nell'ambito dei linguaggi formali, vengono chiamati *simboli* (fondamentali).

Una proprietà caratteristica dei linguaggi formali è che i simboli non possono essere utilizzati in sequenze casuali se si vogliono formare delle frasi (ossia sequenze di simboli portatrici di significato), ma secondo determinate regole.

Possiamo identificare sostanzialmente tre tipologie di regole per formare frasi corrette e ben formulate:

- regole relative ai simboli fondamentali;
- regole relative alla formulazione delle frasi;
- regole relative alla interpretazione delle frasi.

Gli elementi di controllo del singolo simbolo fondamentale sono normalmente forniti dall'*ortografia*. La costruzione della singola parola di un linguaggio obbedisce, infatti, a regole puramente ortografiche.

Gli elementi di controllo della frase, invece, sono normalmente fornite dalle regole *sintattiche*. Si può facilmente ipotizzare una sequenza di parole ortograficamente corrette, ma la cui sintassi risulta errata.

Infine, le regole di interpretazione di una frase sono appannaggio della *semantica*, ovvero dello studio del significato di una frase. Una determinata sequenza di parole può essere ortograficamente corretta, sintatticamente inoppugnabile ma semanticamente errata.

Ad esempio, la frase "*Paolo dorrrme*", contiene un evidente errore ortografico, mentre la frase "*Carlo dorme se svegliarlo non*, non contiene evidenti errori ortografici nelle singole parole, ma è certamente errata dal punto di vista sintattico. Infine, la frase "*E' più veloce il treno o più bianco il latte?*", non contiene né errori ortografici né sintattici ma, oltre ad essere una frase decisamente strana, è sicuramente mal formata dal punto di vista semantico. Ogni singola parola, infatti, risulta essere corretta e così pure la loro sequenza, ma la frase appare priva di significato.

Nonostante quanto appena detto, non è sempre sufficiente formulare frasi corrette dal punto di vista ortografico, sintattico e semantico per attuare una comunicazione efficace. Chi vuole essere sicuro che il proprio messaggio venga colto nella propria pienezza può ricorrere ad ulteriori due strumenti: la *ridondanza* e la *conferma*.

Se vogliamo comunicare in forma ridondante che "*Paolo dorme*", potremmo dire che "*Paolo si è appena appisolato. Non disturbate Paolo che sta dormendo*".

Infine, è vezzo comune agli insegnanti intercalare le loro spiegazioni con numerosi "*E' chiaro? Avete capito?*", ecc. al fine di accertarsi, attraverso una conferma, se effettivamente gli allievi hanno capito la lezione.

Infine, vi è un'ultima accortezza per aumentare l'efficacia della comunicazione, che però è scarsamente utilizzata nella comunicazione verbale: evitare parole foneticamente ambigue.

Si pensi, ad esempio, alle parole *casa* e *caso*. Esse si distinguono solamente per l'ultima lettera. Spesso, durante la normale comunicazione verbale, la frase, come sanno bene gli attori o i comunicatori in generale, cala di intensità. In tal caso è facile confondere le due parole. Alla luce di quanto detto, si confrontino le seguenti due frasi: "*Vado a casa*" e "*Vado a caso*". Se si assume che la parte finale di una frase possa essere "calante", ci si trova di fronte a due

frasi perfettamente corrette dal punto di vista sia ortografico che sintattico e semantico, ma portatrici di due informazioni completamente diverse.

Nelle pagine seguenti vedremo che i sistemi automatici, e più precisamente i sistemi di trasmissione dati, utilizzano tutti gli strumenti appena illustrati, dalle regole ortografiche, a quelle sintattiche e semantiche, alla ridondanza e all'accertamento, badando, infine, ad evitare ambiguità sui simboli. Questo perché una comunicazione efficace nei sistemi di trasmissione dati non è un vezzo, ma questione di importanza fondamentale.

Prima, però, dovremo parlare brevemente di come veicolare l'informazione nel mezzo di comunicazione e tal fine sarà necessario introdurre alcuni concetti teorici.

1.2 La comunicazione seriale

Quando si parla di comunicazione fra sistemi elettronici si deve distinguere fra comunicazioni parallele e comunicazioni seriali.

Si ha una comunicazione parallela quando più linee concorrono contemporaneamente a veicolare un'unica informazione considerata unitaria detta *simbolo*. Dette linee sono chiamate *bus* e sono solitamente formate da linee che veicolano il simbolo vero e proprio e ulteriori linee, sempre facenti parte del bus, che provvedono al sincronismo dell'informazione, indicando, cioè, quando l'informazione è pronta e disponibile alla lettura.

Esempi di comunicazioni parallele sono quelle ottenute attraverso il bus ISA o mediante il vecchio bus Centronics delle stampanti di qualche anno fa.

Il principale vantaggio di un bus parallelo è dato dall'alta velocità di trasmissione che esso può raggiungere, o meglio dall'elevato numero di simboli al secondo che è in grado di veicolare. Tale velocità di trasmissione è chiamata *baud rate* e prende il nome da Émile Baudot inventore del omonimo codice usato in telegrafia.

Quindi un'informazione veicolata con un baud rate di 1200 baud, viene trasmessa alla velocità di 1200 simboli al secondo. Siccome in una trasmissione parallela ogni simbolo è rappresentato da *n bit* (dove solitamente $n = 8$), ogni secondo si trasmettono $1200 \cdot 8 = 9600$ stati logici.

La trasmissione parallela presenta, però, anche qualche svantaggio. Primo fra tutti il costo. Le singole linee rappresentano frequentemente un costo non secondario di cui si deve tener conto.

Un secondo svantaggio (indirettamente collegato al primo) è dato dalle corte distanze solitamente coperte. Non è infatti conveniente dal punto di vista economico avere linee parallele molto lunghe, inoltre, bus paralleli molto lunghi (oltre i 40-50 cm) sono spesso accompagnati da effetti parassiti che ne limitano fortemente la velocità di trasmissione.

Per tali motivi i bus paralleli sono solitamente utilizzati per collegare apparecchiature disposte vicine l'una all'altra e comunque per brevissimi tratti.

Si ha, invece, una comunicazione seriale quando l'informazione (il simbolo) è veicolata mediante un solo filo. Tali modalità di trasmissione hanno lo svan-

taggio di essere più lente rispetto alle trasmissioni parallele, essendo il *baud rate* più basso di circa 8 volte.[1] Presentano, però, due formidabili vantaggi: sono economiche e adatte alle trasmissioni su lunghe distanze.

Anche nel caso delle trasmissioni seriali si parla frequentemente di *bus*. Il termine non è improprio, perché la comunicazione seriale non si avvale quasi mai di una sola linea.

Vi è sempre bisogno, infatti, di almeno una seconda linea di riferimento e spesso anche di una terza linea per poter mantenere separate le comunicazioni in trasmissione da quelle in ricezione. In taluni casi è presente anche una quarta linea di sincronizzazione, che ci porta ad operare una fondamentale distinzione fra le trasmissioni seriali:

- trasmissioni seriali sincrone;
- trasmissioni seriali asincrone.

L'argomento verrà trattato nelle prossime sezioni.

1.2.1 La trasmissione seriale sincrona

Una ulteriore modalità di trasmissione è rappresentata dalla trasmissione sincrona. Essa presuppone che oltre alla linea (o alle linee) ove viene trasmessa l'informazione scorra, fra trasmittente e ricevente, anche una linea di sincronismo, chiamata *clock*. Il clock viene prodotto dal trasmittente ed è responsabile della sicronizzazione del segnale ovvero ha il compito di indicare al sistema ricevente quando il segnale contenente l'informazione è idoneo ad essere campionato.

Tale meccanismo funziona bene se il dispositivo trasmittente e quello ricevente sono posti a distanze contenute, sull'ordine dei centimetri o decine di centimetri. Oltre tali distanze c'è sempre il pericolo di disturbi sulla linea di clock, che potrebbero invalidare totalmente l'informazione ricevuta, per il semplice motivo che verrebbe campionata nel momento sbagliato.

Inoltre, l'aggiunta di un'ulteriore linea comporta problemi di costi se essa dovesse essere eccessivamente lunga.

Il vantaggio di tale soluzione in fase di trasmissione è la possibilità di raggiungere elevate velocità di trasmissione in totale assenza di problemi di sincronismo.

Le linee seriali sincrone sono, per i suddetti motivi, confinate alle comunicazioni seriali fra dispositivi posti sullo stesso circuito stampato o, al massimo, fra schede elettroniche adiacenti e vicine fra loro.

Esempi di tali bus seriali sono il bus I^2C e lo SPI.

[1] Infatti nelle trasmissioni seriali sarebbe più preciso usare l'unità di misura del *bit rate*, che indica la velocità di trasmissione in funzione del singolo bit.

1.2.2 La trasmissione seriale asincrona

La trasmissione asincrona elimina totalmente il problema dei disturbi e dei costi (la linea di clock non esiste!), ma lascia intatto il problema del sincronismo. Il problema viene praticamente risolto, come si vedrà nel prossimo capitolo, dal livello fisico, trasmettendo, insieme all'informazione, dei bit di sincronismo ad ogni byte. In tal modo, se trasmittente e ricevente conoscono con precisione il periodo di bit ed il tipo di codifica di linea utilizzata, è possibile, ogni byte, sincronizzare i rispettivi orologi che fungono in tal modo da clock virtuale.

La comunicazione funziona perfettamente se i due orologi sono sufficientemente precisi, se viene riconosciuto con sufficiente precisione il bit di sincronismo e se la sequenza di bit fra una sincronizzazione e quella successiva non è eccessivamente lunga.
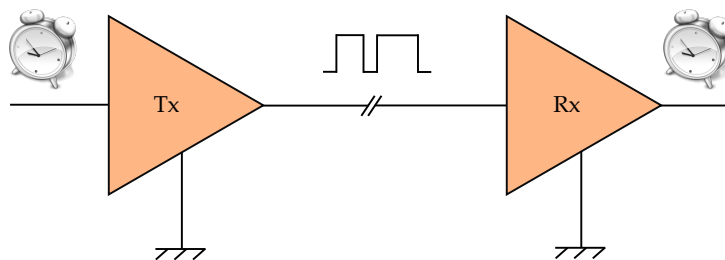
Figura 1.1: La trasmissione seriale asincrona

Questa modalità di trasmissione viene ormai utilizzata in tutte le reti, da quelle locali a quelle che geograficamente si estendono da un continente all'altro. D'ora innanzi, ogni qual volta si parlerà di trasmissione seriale o di comunicazione in generale, si supporrà sempre una trasmissione asincrona.

Siccome la trasmissione seriale asincrona è di particolare importanza nell'economia del presente corso, si aggiungerà qualche ulteriore dettaglio per una sua migliore comprensione. In particolare sul tipo di trasmissione (sbilanciata e bilanciata) e sulle codifiche di linea, ovvero sul come rappresentare il valore logico del singolo bit.

1.2.3 La trasmissione sbilanciata

La trasmissione sbilanciata (*unbalanced*) è indicata per le corte e cortissime distanze e/o quando l'ambiente è sostanzialmente privo di rumore elettrico di fondo. E' pertanto indicata nelle trasmissioni all'interno della singola scheda, ma può estendersi fino alle centinaia di metri, utilizzando determinate precauzioni.[2] E' utilizzata nello standard di trasmissione RS232C. Un esempio di tale tipo di trasmissione è indicato in fig. 1.2 nella pagina seguente.

Essa è caratterizzata da un riferimento di zero, tipicamente la massa del circuito di trasmissione, ed una tensione positiva/negativa rispetto al riferimento.

[2]Lo standard RS232C indica la distanza massima in 70 piedi, ovvero poco più di 20 metri. Col tempo lo standard è stato più volte disatteso e variamente interpretato.

Senza addentrarci nelle diverse codifiche di linea (che verranno analizzate fra breve) il trasmettitore deve utilizzare una determinata codifica per rappresentare gli stati logici 1 e 0. In osservanza a tale codifica quando il trasmettitore vuole trasmettere un 1, pone sulla linea una tensione v_1, ad esempio, positiva rispetto al riferimento di zero, tipicamente +5V oppure +12V. Quando, invece, vuole trasmettere uno 0, pone sulla linea, tipicamente, un valore di tensione v_2 negativo (e simmetrico al precedente) rispetto al riferimento di zero.
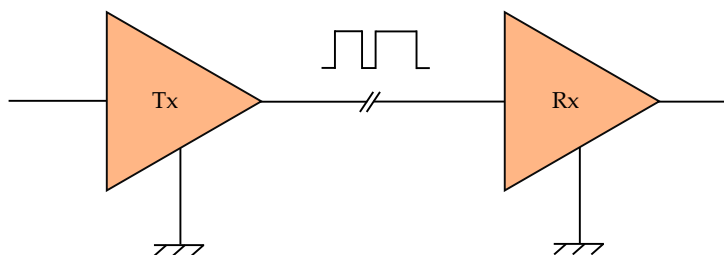
Figura 1.2: La trasmissione sbilanciata

L'andamento della tensione sulla linea di trasmissione rispetto al riferimento di zero può avere l'andamento di fig. 1.3
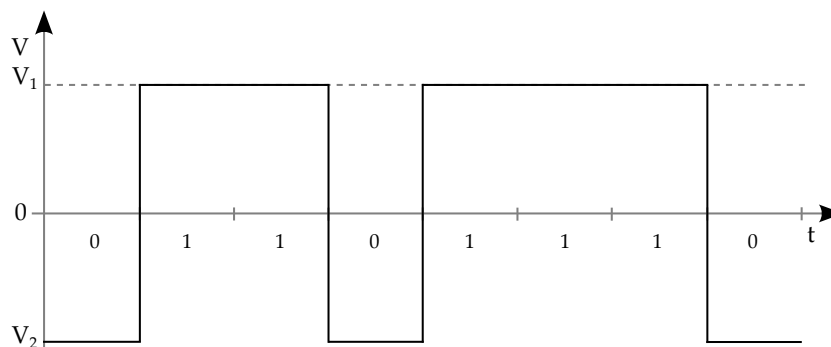
Figura 1.3: Livelli di tensione nella trasmissione sbilanciata

Si noti che l'equivalenza *1 = tensione positiva* e *0 = tensione negativa* non è automatica, ma dipende dagli standard utilizzati.

Si è già accennato al fatto che tale modalità di trasmissione presenta però qualche problema, soprattutto se la distanza fra i due dispositivi è rilevante. In tal caso, infatti, potrebbero esserci problemi di riferimento fra trasmettitore e ricevitore, ovvero le masse delle due apparecchiature potrebbero non essere equipotenziali, oppure potrebbero essere oggetto di disturbi condotti.

La trasmissione sbilanciata è oggidì piuttosto in disuso, causa la lentezza e la sensibilità ai disturbi. E' ancora usata per comunicazioni fra periferiche e l'USART del microcontrollore, quindi per comunicazioni intra-sistema (distanze di qualche decina di centimetri). In caso di lunghe distanze, alte velocità o ambiente rumoroso non ha le caratteristiche adatte ad essere usata.

In tal caso potrebbe essere di maggior aiuto una trasmissione bilanciata.

1.2.4 La trasmissione bilanciata

La trasmissione bilanciata (*balanced*) è indicata per distanze superiori a quelle coperte dalla trasmissione sbilanciata e/o quando l'ambiente è maggiormente disturbato o le velocità di trasmissione devono essere necessariamente alte.

E' pertanto indicata a coprire distanze dell'ordine del chilometro oppure per trasmissioni dell'ordine del Mb/s (può arrivare fino al Gb/s o alla decina di Gb/s). E' utilizzata negli standard di trasmissione RS422, RS485 e i numerosi bus di campo fra i quali il Profibus.

Dal punto di vista strettamente hardware il circuito è leggermente più complesso, come pure il semplice cablaggio fra trasmettitore e ricevitore. Ha però il vantaggio immediato di eliminare il riferimento di massa e di conseguenza anche tutti i problemi derivanti da eventuali differenze di potenziale fra le masse delle apparecchiature. Un esempio di trasmissione bilanciata si ha in fig. 1.4.
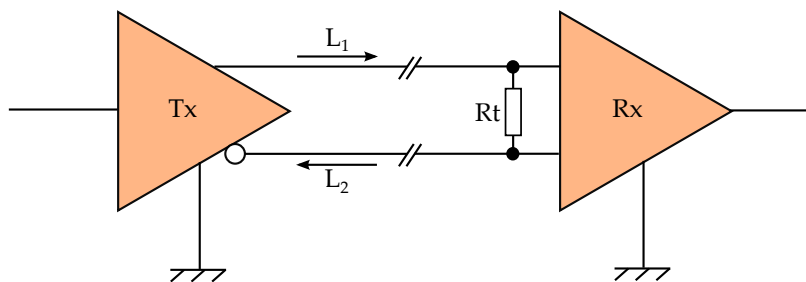
Figura 1.4: La trasmissione bilanciata

La capacità della trasmissione bilanciata a raggiungere alte velocità e lunghe distanze è strettamente legata all'uso di due linee L_1 e L_2 anziché una solamente. Quando il trasmettitore vuole trasmettere un bit a 1, pone la linea L_1 ad una tensione positiva v_1 e la linea L_2 ad una tensione negativa v_2. Le due tensioni v_1 e v_2 sono simmetriche rispetto allo zero e sono dette, per tal motivo, in opposizione di fase.

La differenza di potenziale $v_1 - v_2$ produce in tal caso un segnale positivo pari a $2v_1$. Quando, viceversa, il trasmettitore vuole trasmettere un bit a 0, pone la linea L_1 ad una tensione negativa v_1 e la linea L_2 ad una tensione positiva v_2. La differenza di potenziale $v_1 - v_2$ produce in tal caso un segnale pari a $-2v_1$.

Se un disturbo colpisce l'informazione mentre viaggia sulle linee (vedi la figura 1.5 nella pagina successiva) esso influenza le due linee nello stesso modo e con la stessa polarità. Ciò significa che se si sovrappone al segnale un disturbo positivo, esso viene sommato ad entrambi i segnali.

Il disturbo presente sulla linea L_2 verrà però sottratto a quello presente sulla linea L_1, eliminando di fatto il segnale indesiderato e mantenendo intatta l'informazione.

Eseguire la differenza fra i due segnali, oltre ad eliminare i disturbi presenti sulla linea, permette anche di essere totalmente svincolati rispetto al riferimen-

to assoluto (la massa) e di valutare solamente il livello di tensione delle due linee di comunicazione.
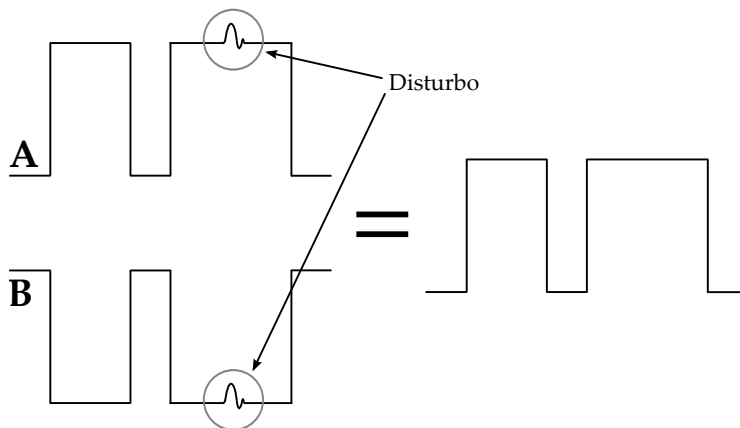
Figura 1.5: Eliminazione del disturbo nella trasmissione bilanciata

Al termine della linea viene posta una resistenza di terminazione, avente il compito di evitare riflessioni in linea. Tipicamente tale valore è di $100 - 120\,\Omega$.

Vedremo che il presente tipo di trasmissione utilizza ulteriori accorgimenti che permettono il raggiungimento di velocità di comunicazione molto alte. Prima, però, è necessario analizzare le codifiche di linea, ovvero come modificare i livelli di tensione per codificare l'informazione.

1.3 Le codifiche di linea

Quando si devono trasmettere dei dati numerici lungo un cavo elettrico si hanno sostanzialmente due possibilità: o l'informazione viene modulata (*broadband* o banda larga), oppure no (*baseband* o banda base).

Nel primo caso si parte da un segnale noto (ad esempio un segnale sinusoidale), detto portante, e lo si modifica in ampiezza o in frequenza, in modo che le informazioni siano contenute nelle modifiche di ampiezza o di frequenza.

Nel secondo caso si invia l'informazione sul cavo senza modularla. In tal caso si deve scegliere una codifica di linea, che abbia il compito di codificare l'informazione. Essa deve avere alcune determinate caratteristiche:

- essere il più possibile immune ai disturbi;
- permettere una facile rilevazione degli errori;
- raggiungere alte velocità di trasferimento lungo il cavo;
- bassa densità spettrale.

L'informazione che deve essere codificata è organizzata in *bit* (contrazione di *Binary digiIT*) che rappresentano gli stati logici 1 (vero) e 0 (falso).

Dal punto di vista fisico tali stati sono rappresentati da due differenti livelli di tensione o dalle relative transazioni necessarie a raggiungere tali livelli. Se

dovessimo rappresentare su assi cartesiani l'informazione che scorre lungo il cavo, porremmo gli stati logici, ovvero le tensioni, sull'ordinata, mentre sull'ascissa porremmo il tempo, ovvero la durata di ciascun bit, chiamata periodo di bit.

Tra le codifiche di linea maggiormente usate ci sono la NRZ, la Manchester, la Manchester differenziale, la 4B/5B, la 5B/6B e la 8B/10B. Tra quelle di interesse puramente storico si possono elencare la NRZI e la RZ, .

1.3.1 La codifica NRZ

NRZ significa *N*on *R*eturn to *Z*ero. Lo stato logico 1 è rappresentato da un livello di tensione alto, mantenuto tale per tutto il periodo di bit. Questa particolarità dà il nome alla codifica.

Lo stato logico 0 è rappresentato da un livello di tensione basso, mantenuto tale per tutto il periodo di bit. Un esempio di tale codifica è illustrato in fig. 1.6.
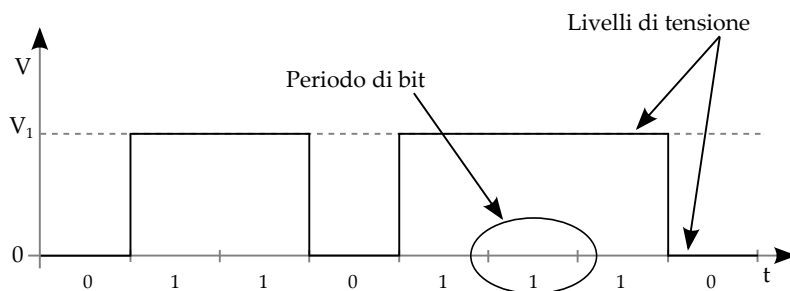
Figura 1.6: La codifica NRZ

Si suppone che il tempo t_0 sia posto all'origine degli assi e che esso cresca sull'asse dell'ascissa. I primi 8 bit hanno quindi valore binario 01101110, ovvero valore esadecimale 0x6E. La codifica ha il difetto che lunghe sequenze di 1 oppure di 0 possono portare fuori sincronismo il segnale.

Uno dei possibili modi per minimizzare tale difetto consiste nell'usare delle particolari tabelle di transcodifica denominate codifica 4B/5B oppure codifica 5B/6B che trasformano gruppi di 4 bit dell'informazione originale in gruppi di 5 bit (4B/5B) oppure gruppi di 5 bit dell'informazione originale in gruppi di 6 bit (5B/6B). Ciò implica un incremento della larghezza di banda rispettivamente del 25% e del 20%, ma una maggior sicurezza, dato che vengono eliminate sequenze troppo lunghe di 0 o di 1.

1.3.2 La codifica NRZI

L'acronimo significa *N*on *R*eturn to *Z*ero *I*nverted. Lo stato logico 0 è rappresentato dalla permanenza del livello precedente, mentre lo stato logico 1 dal

cambio di livello. Ciò comporta che lunghe sequenze di 0 possono portare fuori sincronismo il segnale, mentre lunghe sequenze di 1 creano un fronte di discesa o di salita ad ogni bit, aiutando a mantenere il sincronismo. I problemi di sincronismo rispetto ad una codifica NRZ sono quindi ridotti del 50%.

Nonostante ciò, la codifica NRZ risulta tuttora molto più usata rispetto alla NRZI. I motivi sono principalmente legati al fatto che la NRZ rimane la codifica con l'uso più efficente della banda e che i problemi di sincronismo sono fortemente limitati dall'uso di protocolli orientati al carattere piuttosto che al bit. In tal modo, l'inizio di un carattere e la sua fine sono sempre indicati da livelli logici differenti, che sincronizzano comunque il segnale.

Un esempio di tale codifica è rappresentato in fig. 1.7.
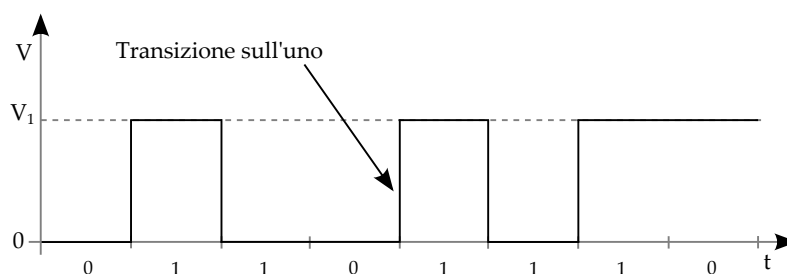
Figura 1.7: La codifica NRZI

A parità di codice binario, la codifica appare molto diversa rispetto alla NRZ.

1.3.3 La codifica RZ

La codifica RZ (*R*eturn to *Z*ero) prevede il ritorno a zero, qualsiasi sia il livello, nella seconda metà del periodo di bit. Lo stato logico 1 è quindi rappresentato da un livello 1 nella prima metà del periodo di bit e da un livello 0 nella seconda metà.
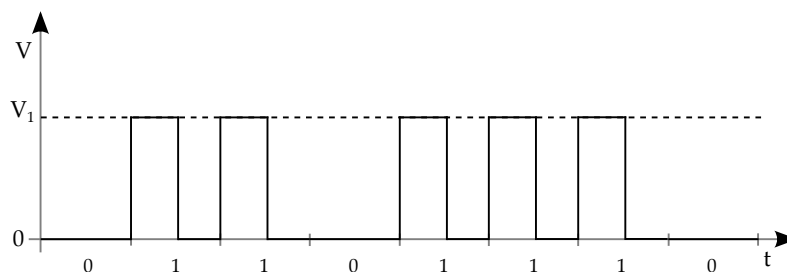
Figura 1.8: La codifica RZ

Lo stato logico 0 è rappresentato da un livello zero per l'intero periodo di bit. In fig. 1.8 a fronte è rappresentato l'andamento della codifica per il solito codice binario.

Anche in questo caso, come per la codifica NRZI, il sincronismo può essere perso in caso di lunghe sequenze di 0. Inoltre, siccome la codifica RZ usa metà periodo, essa abbisogna del doppio della banda rispetto alle codifiche precedenti.

1.3.4 La codifica Manchester

La codifica Manchester ha la particolarità di presentare una transizione per ogni periodo di bit. Lo stato logico 1 è rappresentato da una transizione dal livello basso al livello alto che avviene nella metà del periodo di bit. Lo stato logico 0 è rappresentato da una transizione dal livello alto al livello basso che avviene nella metà del periodo di bit. Tale codifica ha il pregio di mantenere sempre e comunque il sincronismo, essendoci una transizione per ogni bit. Un esempio è rappresentato in fig. 1.9.
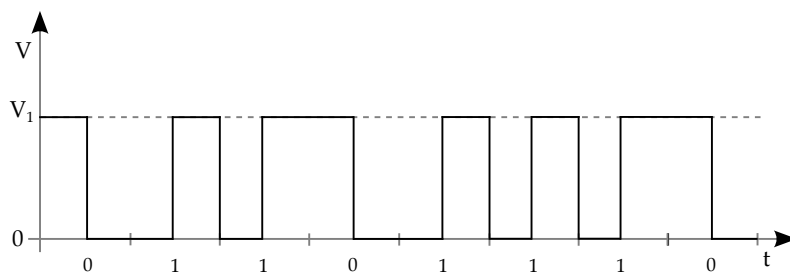
Figura 1.9: La codifica Manchester

La codifica Manchester ha lo stesso difetto della RZ, necessitando del doppio della banda passante rispetto alle precedenti due codifiche è, però, popolare perché utilizzata nello standard IEEE 802. Quando sono necessarie alte velocità di comunicazione, è abbinata ad una codifica 8B/10B.

1.3.5 La codifica Manchester differenziale

Una versione un po' più complicata della codifica Manchester è la Manchester differenziale. Anch'essa divide ogni periodo di bit in due parti uguali e quindi presenta lo stesso difetto, in termini di larghezza di banda occupata, della Manchester.

Lo stato logico 1 è rappresentato dall'assenza di transizione (o un fronte di salita o un fronte di discesa) all'inizio del bit, mentre lo stato logico 0 è rappresentato dalla presenza di transizione. A metà del periodo di bit vi è sempre

una transizione di livello, sia per lo stato logico 1 che per lo stato logico 0, che permette di mantenere la sincronizzazione.

Questo tipo di codifica necessita di un hardware più complesso per essere realizzata, ma garantisce una immunità al rumore superiore alla codifica Manchester. Ciò è garantito anche da una componente continua del segnale complessivo vicina agli 0V, dato che i due livelli del segnale sono simmetrici rispetto allo zero.

Il livello alto è tipicamente compreso fra 3V e 4.5V, mentre il livello basso è compreso fra -3V e -4.5V.

Un esempio di codifica Manchester differenziale è dato in fig. 1.10. Si fa notare che, come per gli altri tipi di codifica, i due livelli sono rapresentati con V_1 e V_2.
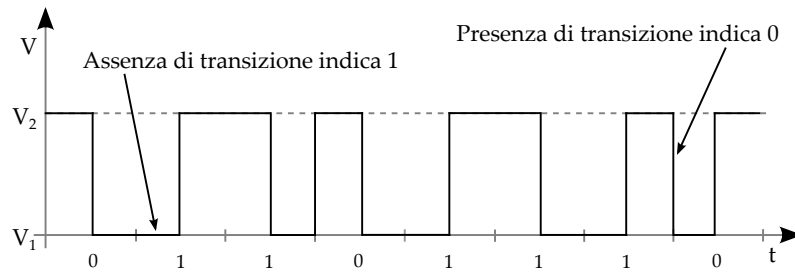
Figura 1.10: La codifica Manchester differenziale

Lo standard IEEE 802 utilizza tale tipo di codifica nella IEEE 802.5, detta Token Ring.

1.3.6 La codifica 4B/5B

Nelle sezioni precedenti si è parlato di codifica 4B/5B e di codifica 8B/10B. Entrambe sono utilizzate nello standard IEEE 802, popolarmente denominato Ethernet, per cui si ritiene utile fornire qualche dettaglio. La codifica 4B/5B

$4B_L$	$5B_L$	$4B_H$	$5B_H$
0000	11110	1000	10010
0001	01001	1001	10011
0010	10100	1010	10110
0011	10101	1011	10111
0100	01010	1100	11010
0101	01011	1101	11011
0110	01110	1110	11100
0111	01111	1111	11101

Tabella 1.1: Codifica 4B/5B

(vedi tabella 1.1) è definita nello standard IEEE 802.3u del 1995 che regola la trasmissione 100BASE-TX, nota come Gigabit ethernet su cavi UTP di categoria 6. Essa trasforma un codice di 4 bit in un codice di 5 bit. Questa trasformazione presenta vantaggi e svantaggi.

Lo svantaggio prevalente è palese: serve il 25% di banda in più se si vuole rappresentare un codice da 4 bit mediante 5 bit. Vi è, però, anche un vantaggio immediato: vengono interrotte le lunghe sequenze di zeri come pure quelle di uni. Inoltre, il passaggio attraverso la 4B/5B garantisce uniformità in DC del segnale e uno spettro più adeguato alle alte velocità.

1.3.7 La codifica 8B/10B

La codifica 8B/10B fornisce gli stessi vantaggi e gli stessi svantaggi della 4B/5B.

Siccome vengono utilizzati 10 bit per codificare un codice da 8, è possibile eliminare le sequenze di 5 bit consecutivi mantenendo un ottimo equilibrio in DC. Ciò permette di trasmettere il flusso di bit disaccoppiandolo mediante trasformatore ethernet senza ulteriori interventi hardware. Tale tecnica è detta *accoppiamento capacitivo* e può essere usata se il singolo carattere è perfettamente bilanciato. Ciò si raggiunge nel seguente modo.

Gli 8 bit di partenza sono divisi in due gruppi da 5+3 bit. Al primo gruppo viene associata una sequenza di 6 bit, mentre al secondo una sequenza di 4 bit.

A ciascun gruppo possono essere associate a delle sequenze di input delle sequenze di output perfettamente bilanciate, aventi cioè lo stesso numero di 0 e di 1. Ad esempio, alla sequenza di input 001 (sbilanciata) è associata una sequenza di output 1001 (bilanciata).

Non ci sono, però, sufficienti combinazioni per bilanciare tutte le sequenze di output. Si ricorre allora alla tecnica denominata *bounded disparity*. Ogni sequenza di input è associata a due sequenze di output: una con uno 0 in più ed una con un 1 in più. Se il gruppo precedente era sbilanciato con uno 0 in più, si sceglie la sequenza con un 1 in più. In tal modo la sequenza risulta essere bilanciata sui 10 bit.

1.4 L'*USART*

L'acronimo *USART* significa *Universal Synchronous Asynchronous Receiver Transmitter*, ovvero Ricetrasmettitore Sincrono/Asincrono Universale. Si tratta di un dispositivo hardware (più precisamente un circuito integrato) che a partire dagli anni sessanta ha affiancato il microprocessore e poi il microcontrollore per sgravarlo dal compito della ricetrasmissione sincrona o asincrona lungo i canali seriali.

I primi USART sono stati prodotti dalla *Intel* (8251), dalla *National Semiconductors* (16550) e dalla *Zilog* (SIO). Possedevano un bus di interfacciamento verso il microprocessore e si occupavano autonomamente di eseguire le seguenti azioni:

Trasmissione

- convertire il dato parallelo in dato seriale;
- avvertire il microprocessore che il dato è stato convertito da parallelo a seriale;
- aggiungere i bit di *framing*;
- trasmettere il carattere seriale secondo la velocità selezionata.

Ricezione

- ricevere il carattere seriale secondo la velocità selezionata;
- convertire il dato da seriale a parallelo;
- verificare la correttezza del dato ricevuto;
- togliere i bit di framing;
- avvertire il microprocessore che c'è un nuovo dato pronto da leggere.

Da allora le differenze introdotte dai moderni USART sono state minime. Tutt'ora la trasmissione inizia con la scrittura del dato da trasmettere, da parte del microprocessore o del microcontrollore, nel registro di trasmissione dell'USART. Da questo momento in poi l'USART procede autonomamente nella trasmissione.

Il dato (a 8 o a 16 bit, dipende dal sistema) viene innanzitutto spostato dal registro parallelo di trasmissione al registro seriale di trasmissione, dove vengono aggiunti i bit di *framing*, per delimitare o *incorniciare* il dato originale. Appena il dato è stato trasferito dal registro parallelo di trasmissione a quello seriale, l'USART avverte il micro, in modo che esso possa già preparare il nuovo eventuale dato da trasmettere.

Poi, se richiesto, viene calcolato ed aggiunto al dato seriale un bit di ridondanza, per aumentare la sicurezza della trasmissione.

Ciò fatto inizia la trasmissione seriale vera e propria. L'USART pone sulla propria linea di uscita il primo bit del *frame*, detto bit di *Start* e poi a seguire gli 8 o 16 bit di dato, a partire dal meno significativo.

Trasmesso l'ultimo bit di dato viene trasmesso l'eventuale bit di ridondanza ed infine uno o più bit di fine *frame*.

Dall'altro capo del filo un altro USART riceve lo *stream* di bit attraverso il registro seriale di ricezione e, conoscendo la struttura del singolo carattere, ricostruisce il dato parallelo e lo trasferisce nel registro parallelo di ricezione. Prima di avvertire il micro che c'è un nuovo dato pronto vengono effettuati una serie di controlli per verificare se il carattere ricevuto è affetto da errore oppure no. Dopodiché viene avvertito il microprocessore.

Quest'ultimo ha il compito di leggere il registro parallelo di ricezione prima dell'arrivo di un nuovo carattere.

Si noti che l'USART tratta in uscita al registro di trasmissione seriale ed in ingresso al registro di trasmissione seriale livelli di tensione cosidetti *TTL*. Significa che uno zero logico è rappresentato mediante un livello di tensione pari a zero, e un uno logico mediante un livello di tensione pari alla tensione di alimentazione dell'USART.

La codifica di linea utilizzata solitamente dagli USART è una NRZ.

1.4.1 I parametri di trasmissione

Nella sezione precedente è stata illustrata sommariamente la trasmissione seriale effettuata dall'USART. Si è visto che la trasmissione è regolata da una serie di parametri di comunicazione ai quali si è appena accennato:

- la velocità di trasmissione;
- i bit di *framing*;

Tali parametri devono essere noti e condivisi fra i dispositivi che comunicano fra loro, pena l'accumulo di errori di comunicazione. La scelta di tali valori dipende naturalmente dal contesto e dal grado di affidabilità che si vuole ottenere.

La velocità di una trasmissione (*baud rate* o, più correttamente, *bit rate*) sbilanciata con codifica NRZ può assumere solitamente i seguenti valori, espressi in bit al secondo:

110	300	600	1200	2400	4800	9600
14400	19200	38400	57600	115200	128000	256000

La scelta della velocità di trasmissione dipende essenzialmente dalla distanza che si vuole coprire in trasmissione e dal tipo di trasmissione usata (sbilanciata o bilanciata). Una trasmissione sbilanciata a 25m, sarebbe bene non superasse i 38400bps, mentre una trasmissione bilanciata può superare tranquillamente la decina di Mbps coprendo una distanza superiore ai 100m.

I bit di *framing* sono sostanzialmente 3:

- il bit di start;
- il bit di parità;
- il bit di stop.

Il bit di start è il primo bit trasmesso e vale sempre 0. Il bit di parità (bit di ridondanza) serve ad aumentare l'affidabilità del sistema aumentando la ridondanza. La prossima sezione sarà dedicata a tale argomento. Il bit di stop può essere in numero di 1, 1.5 oppure 2. Anche tale aspetto, curioso per certi aspetti, verrà esaminato nelle prossime sezioni.

Un esempio di parametri di scelta dei parametri di trasmissione potrebbe, quindi, essere il seguente:

9600bps, Nessuna parità, 1 bit di Stop

Si noti che non è necessario dichiarare il bit di start, perché implicito e costante.

1.4.2 La parità

Si è parlato di ridondanza e di parità nella sezione precedente. L'argomento verrà analizzato in maggior dettaglio nelle presenti righe.

Come evidenziato nella sezione 1.1.1 a pagina 3 la ridondanza è un utile mezzo per aumentare l'affidabilità della comunicazione, permettendo all'ascoltatore di valutare se parte del messaggio è andato perso.

Il bit di parità assolve esattamente a tale compito. Il meccanismo consiste nell'aggiungere un bit all'informazione, il cui valore viene determinato secondo una semplice regola nota sia a chi trasmette che a chi riceve. In fase di ricezione si valuta se il bit aggiunto rispetta la regola oppure no. In caso contrario si può giustamente ipotizzare un errore in fase di trasmissione.

La parità può essere di sostanzialmente di tre tipi:

- parità pari;
- parità dispari;
- nessuna parità.

Si supponga che gli utenti coinvolti nella comunicazione abbiano scelto una parità pari. Prima di spedire il carattere, l'USART conta tutti gli "1" presenti nel solo dato (8 oppure 16 bit) e aggiunge o meno un "1" nel bit di parità in modo tale la somma totale degli "1" (dato + parità) sia pari.

Il meccanismo è analogo se la parità scelta è dispari.

Se la parità non viene utilizzata detto bit risulta proprio mancante.

1.4.3 La trasmissione del carattere

Ora è possibile descrivere nel dettaglio come avviene la trasmissione e la ricezione di un carattere seriale. Si supponga, ad esempio di voler trasmettere il carattere "F" (01000110_2) alla velocità di 9600bps, con 1 bit di Stop e Parità Dispari, come illustrato graficamente in fig.1.11.
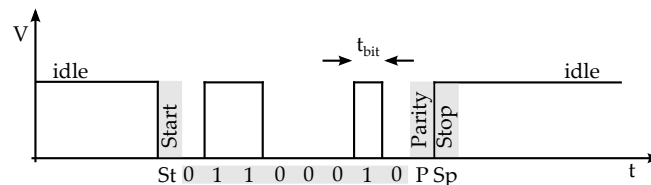
Figura 1.11: Carattere da trasmettere

Innanzi tutto va valutata la bontà della sincronizzazione. A tal fine si devono conoscere tre parametri fondamentali, oltre alla velocità di trasmissione:

- la frequenza di campionamento della linea da parte del ricevitore;
- il numero di campionamenti per ciascun bit;
- la tolleranza dichiarata su ciascuna velocità di comunicazione (trasmettitore e ricevitore).

I suddetti parametri caratterizzanti l'USART sono importanti per il seguente motivo.

L'USART ricevente campiona ad una frequenza finita la linea per verificare se stanno arrivando dei caratteri. In stato di non trasmissione tale linea è posta a "1" logico. Quando arriva il primo bit di un carattere, questo sarà un bit

di Start (stato logico "0"), e il ricevitore si sincronizzerà proprio sul fronte di discesa di tale bit.

E' quindi molto importante conoscere la frequenza di campionamento della linea, perché da essa dipende la precisione dell'aggancio del fronte di discesa dello Start. Quindi un primo passo da effettuare consiste nel documentarsi su tale frequenza. Essa dipende dall'USART utilizzato in ricezione.

In assenza di dati a tal proposito si può assumere una frequenza di campionamento di 16 volte superiore a quella di comunicazione selezionata, che rappresenta un dato piuttosto frequente.

Un secondo parametro da tener presente è dato dal numero di campionamenti effettuati su ciascun bit. Normalmente gli USART eseguono un solo campionamento per ciascun bit, ma alcuni dispositivi campionano il bit di Stop più di una volta. Alcuni USART della Microchip, ad esempio, utilizzati come periferiche interne a certi microcontrollori, campionano l'ultimo bit di Stop tre volte (il motivo di ciò verrà esaminato fra poco).

Un ultimo parametro è dato dalla tolleranza sulla velocità di comunicazione dei due USART. Una cattiva sincronizzazione, infatti, non è solo data dal fatto che i fronti dei clock non siano sincroni, ma anche e soprattutto dal fatto che la frequenza dei due clock sia leggermente diversa.

Si supponga, a titolo d'esempio, che l'USART riceventi campioni ad una frequenza 16 volte superiore alla velocità di comunicazione; che tutti i bit, compreso quello di Stop, vengano campionati una sola volta; che la tolleranza sulla velocità di comunicazione dell'USART trasmittente sia del 5% e che quella del ricevente sia del 4% (entrambe comunque pessime: si cerca di restare sotto l'1%).

La domanda che ci si pone è la seguente: quanto affidabile è una comunicazione con queste caratteristiche?

Si deve ipotizzare il *worst case*, ossia il peggiore dei casi possibili. Si deve cioè ipotizzare che il t_{bit} del trasmettitore valga, ad esempio:

$$t_{TXbit} = \frac{1}{9600} - 5\% = 104,17 \cdot 10^{-4} - \frac{5}{100} \cdot 104,17 \cdot 10^{-4} \approx 99,96\ \mu s \qquad (1.1)$$

Per contro, si deve ipotizzare che il t_{bit} del ricevitore valga:

$$t_{RXbit} = \frac{1}{9600} + 4\% = 104,17 \cdot 10^{-4} + \frac{4}{100} \cdot 104,17 \cdot 10^{-4} \approx 108,34\ \mu s \qquad (1.2)$$

Il campionamento del fronte di discesa del bit di Start potrà avvenire, nel peggiore dei casi, con un ritardo t_c pari ad un sedicesimo di t_{RXbit}, ovvero $t_c = 6,77\ \mu s$.

Inoltre, il bit di Start non verrà campionato esattamente a metà del t_{bit}, ossia dopo $99,96 : 2 = 49,98\ \mu s$ dal rilevamento del fronte di discesa del bit di Start ma dopo $6,77 + 108,34 : 2 = 60,94\ \mu s$,[3] con un ritardo complessivo di $60,94 - 49,98 = 10,96\ \mu s$.

Il bit 0 del contenuto informativo verrà campionato dopo ulteriori $108,34\ \mu s$ e non dopo $99,96\ \mu s$, il che significa che il campionamento del bit 0 avverrà

[3] Tale errore è dovuto al fatto che il ricevitore calcola la metà del t_{bit} secondo il proprio orologio interno, che prevede un t_{bit} di $108,34\ \mu s$ e non di $49,98\ \mu s$.

dopo $6,77 + 54,17 + 108,34 = 169,28\ \mu s$ dal fronte di discesa del bit di Start invece che dopo $49,98 + 99,96 = 149,94\ \mu s$, ossia con un ritardo complessivo di $169,28 - 149,94 = 19,34\ \mu s$.

Come si vede il ritardo aumenta. Ciò significa che il campionamento, lentamente, deriva.

Continuando il ragionamento, il bit 1 verrà campionato dopo $277,62\ \mu s$ dal fronte di discesa del bit di Start invece che dopo $249,90\ \mu s$. Il bit 2 verrà campionato dopo $385,96\ \mu s$ dal fronte di discesa del bit di Start invece che dopo $349,86\ \mu s$. Il bit 3 verrà campionato dopo $494,30\ \mu s$ dal fronte di discesa del bit di Start invece che dopo $449,82\ \mu s$. Il bit 4 verrà campionato dopo $602,64\ \mu s$ dal fronte di discesa del bit di Start invece che dopo $549,78\ \mu s$, con un ritardo di $52,86\ \mu s$. Tale ritardo è superiore alla metà del t_{bit} del trasmettitore, ossia $49,98\ \mu s$, per cui, invece di campionare il bit 4, verrà campionato erroneamente il bit 5.
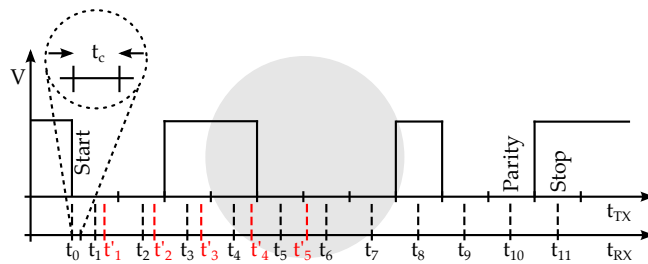
Figura 1.12: Campionamenti sul carattere ricevuto

In realtà l'USART non è in grado di accorgersi di tali errori finché non calcola la parità, che risulterà effettivamente errata. Ma l'USART non rileverà solamente un errore di parità (che potrebbe anche sfuggire se gli "1" sono posizionati in maniera sfavorevole) ma soprattutto verrà campionato uno "0" come bit di Stop e tale condizione sarà sufficiente per far capire all'USART che vi sono problemi di sincronizzazione.

1.4.4 Gli errori di ricezione

Gli errori rilevati sul singolo carattere dall'USART in ricezione possono sostanzialmente essere di tre tipi:

- *Framing error*;
- *Parity error*;
- *Overrun error*.[4]

L'errore di *framing* consiste nell'errata rilevazione del bit di Stop. Una scarsa sincronizzazione può produrre tale errore come effetto finale.

A tal proposito si coglie l'occasione per illustrare il significato dei 2 bit di Stop o del curioso 1.5 bit di Stop. Non si tratta di un goffo modo di accomodare

[4]Si noti che molti USART non supportano la rilevazione automatica degli errori di parità e *overrun*.

piccoli errori di *framing*, come si potrebbe pensare: una siffatta soluzione sarebbe da censurare come poco seria. Si tratta, invece, della possibilità di introdurre un ulteriore allungamento del carattere e valutare meglio la sincronizzazione. Un buon uso di tale opzione potrebbe consistere nel testare la comunicazione con 2 bit di Stop, ma di effettuarla poi con 1 solo bit di Stop.

La scelta 1.5 significa che immediatamente dopo il secondo campionamento del bit di Stop ci si deve subito attendere il fronte di discesa del bit di Start. Anche questa soluzione ha senso in fase di test piuttosto che in fase di release. Oggidì, comunque, quasi nessun USART supporta più tale selezione.

Rilevare, quindi, un *framing error* significa sempre essere mal sincronizzati fra i due dispositivi.

In tal caso una possibile scelta consiste nell'abbassare la velocità di comunicazione, dato che gli errori introdotti nel *baud rate* variano solitamente da velocità a velocità (se il quarzo usato non è appositamente pensato per le trasmissioni seriali, come ad esempio i quarzi a 2457600Hz, che introducono tolleranze teoriche nulle).

Il compito del bit di parità è evidente. Coll'aumentare dell'affidabilità delle comunicazioni, però, si tende a non usare più il bit di ridondanza sul singolo carattere, per cui molti USART non implementano più la rilevazione automatica di detto errore. Viene semplicemente permessa la trasmissione di un generico nono bit che potrà essere interpretato come parità, ma sarà compito del microprocessore elaborarne i risultati.

Anche l'errore di *overrun* non è più molto supportato negli USART moderni. Si ha *overrun* quando il sistema ricevente non riesce a leggere con sufficiente velocità i caratteri che arrivano sulla linea di comunicazione. Ciò poteva avvenire, in passato, quando il microcontrollore o microporocessore era impegnato in altri servizi e non aveva modo di leggere regolarmente la linea seriale. Qualche USART possedeva un piccolo registro FIFO a 2 o 3 livelli, per cui si riusciva a "parcheggiare" fino a tre caratteri in ricezione prima di leggerli.

La gestione automatica, però, di tale errore non è più sentita come una priorità, dato l'aumento vertiginoso delle velocità di elaborazione dei micro moderni. Si tenga anche conto della notevole differenza di velocità fra elaborazione interna di un micro (anche i microcontrollori che non vantano primati particolari sono in grado di elaborare 10 milioni di istruzioni al secondo) e quella di comunicazione seriale (al massimo qualche migliaia di caratteri al secondo).

Si tornerà ancora pesantemente sull'argomento e si vedrà che sono stati elaborati diversi accorgimenti per rendere via via più affidabili le comunicazioni seriali, estendendo i controlli dal singolo carattere al singolo pacchetto dati, non notevoli benefici in termini di funzionalità.

Nelle prossime sezione si vedrà, comunque, che l'efficienza di una comunicazione non dipende solamente dalla sua affidabilità, ma anche e soprattutto da altri fattori.

1.5 La rilevazione degli errori

Benché si operi con grande cura per assicurare una trasmissione affidabile, non si può ridurre a zero la probabilità di errore.

Nei bus di tipo parallelo, diafonia e campi esterni disturbano il segnale. Nelle trasmissioni seriali, i disturbi sono dovuti principalmente ad interferenze elettromagnetiche (EMI) dovute ad apparecchiature di potenza, relè e correnti di massa. Nelle trasmissioni via radio la qualità del segnale può essere deteriorata da temporali, tempeste solari, pioggia, ostacoli.

Le trasmissioni numeriche sono molto più sensibili agli errori di trasmissione che non l'udito umano. Un rumore tipo "clic" su una linea telefonica praticamente non disturba la conversazione, ma se la stessa linea viene usata per trasmettere dati con un modem a 50Kbit/s, il disturbo rovinerà almeno qualche centinaio di bit.

In realtà, anche nel caso del segnale vocale viene persa all'incirca la stessa quantità di informazione, ma il linguaggio umano presenta molta ridondanza e il cervello è un ottimo decodificatore in grado di ricostruire la maggior parte dell'informazione perduta.

Nelle prossime sezioni si formalizzerà la probabilità che una trasmissione di n bit ha di generare un errore e poi verranno esaminati diverse modalità e tecniche per rilevare o addirittura correggere l'errore di comunicazione.

1.5.1 La probabilità di errore

Si supponga una trasmissione di n bit, ciascuno dei quali può essere affetto da errore con probabilità p.

Se più eventi, indipendenti ed equiprobabili, hanno luogo, le relative probabilità vanno moltiplicate fra loro. Quindi se esiste probabilità p che *un* bit (su n) sia affetto da errore, allora la probabilità che *due* bit (su n) possa essere errata vale p^2.

Analogamente la probabilità sale a p^k se si ipotizzano k bit errati.

Quindi la probabilità che k bit siano affetti da errore e che i restanti $n-k$ siano *error-free* e siccome i k bit possono presentarsi in $\binom{n}{k}$ modi differenti si ha che la probabilità di errore di k bit errati su n bit trasmessi vale:

$$p_n(k) = \binom{n}{k} p^k (1-p)^{n-k} \tag{1.3}$$

e siccome in una trasmissione dati solitamente si può assumere che $p \ll 1$ si ottiene:

$$p_n(k) \approx \binom{n}{k} p^k \tag{1.4}$$

ovvero

$$p_n(k) \approx \frac{n!}{k!(n-k)!} p^k \tag{1.5}$$

1.5.2 L'errore sul carattere

Si è visto nella sezione 1.4.3 che un carattere è formato solitamente da 11 bit: 1 bit di start più 8 bit di dato più 1 bit di parità più 1 bit di stop. Il *bit error rate*, ovvero il tasso di errore di una trasmissione seriale, dovuto principalmente al rapporto segnale/rumore, ad esempio su doppino telefonico, vale circa 10^{-4}, ossia un errore ogni 10000 bit trasmessi.

La domanda che ci si pone è quale sia il tasso di errore di un carattere trasmesso con bit di parità.

Innanzi tutto va sottolineato che trasmettere una parità pari o dispari non modifica il tasso di errore.

Siccome il bit di parità rileva tutti gli errori dispari nel carattere e siccome il controllo dei bit di start e di stop viene eseguito direttamente dall'USART, sul singolo carattere si possono avere solo 2, 4, 6 oppure 8 bit di errore.

La probabilità che ciò avvenga vale:

$$ber_2 = \frac{b!}{k!(b-k)!}p^k = \frac{9!}{2!(9-2)!}10^{-8} = \frac{362880}{10080}10^{-8} = 36 \cdot 10^{-8}$$

$$ber_4 = \frac{b!}{k!(b-k)!}p^k = \frac{9!}{4!(9-4)!}10^{-16} = \frac{362880}{2880}10^{-16} = 126 \cdot 10^{-16}$$

$$ber_6 = \frac{b!}{k!(b-k)!}p^k = \frac{9!}{6!(9-6)!}10^{-24} = \frac{362880}{4320}10^{-24} = 84 \cdot 10^{-24}$$

$$ber_8 = \frac{b!}{k!(b-k)!}p^k = \frac{9!}{8!(9-8)!}10^{-32} = \frac{362880}{40320}10^{-32} = 9 \cdot 10^{-32}$$

dove p è la probabilità di errore sul singolo bit; k rappresenta il numero di bit (pari) di errore nel singolo carattere e b sono i bit del carattere che possono essere affetti da errore. Dagli 11 bit complessivi del carattere vanno quindi tolti il bit di start ed il bit di stop che sono direttamente controllati dall'USART.

Si tenga anche presente che l'ipotesi fatta è di usare un semplice doppino telefonico, che assicura uno standard qualitativo piuttosto basso.

Si nota, comunque, che la probabilità che vi siano 4, 6 oppure 8 errori sul singolo carattere (che non vengono quindi rilevati dal meccanismo di parità) sono piuttosto basse.

La probabilità che su un singolo carattere vi siano 2 errori, però, non è del tutto trascurabile. Ogni 2.8 milioni di caratteri vi è la possibilità che uno sia errato e non rilevato dalla parità.

Pur essendo un simile tasso di errore non altissimo, non può comunque essere accettato. Al fine di minimizzare ulteriormente la comunicazione, vengono adottati ulteriori accorgimenti.

1.5.3 La parità longitudinale

Si supponga di voler trasmettere un breve testo e di voler ricercare un metodo che permetta un'affidabile ricerca degli errori da parte del ricevente. Si supponga anche di ritenere sufficiente lo strumento della ridondanza, per cui si cercherà di elaborare una soluzione in tal senso.

Si è visto che la parità sul carattere rileva tutti gli errori in numero dispari che si presentano sul singolo carattere: se un solo bit del carattere viene alterato rispetto all'originale, la parità (sia essa pari o dispari) lo rileva immancabilmente.

Se vengono alterati 3, 5 o 7 bit del singolo carattere (ossia un numero dispari di bit), la parità rileva ancora l'errore, senza poter evidenziare, però, quanti bit sono stati modificati.

La parità sul carattere è solamente in grado di rilevare errori nella ricezione, non di indicare il numero dei bit falsati. Comunque, sapere che il carattere arrivato è errato può essere sufficiente per una corretta rcomunicazione.

Se, però, il numero di bit alterati è pari, allora l'errore non viene rilevato, e la probabilità che ciò accada non è trascurabile.

Si supponga, allora, di introdurre un secondo tipo di parità, detto *longitudinale* o LRC (*Longitudinal Redoundancy Check*), che consiste in un byte aggiuntivo a fine testo (vedi fig. 1.13).

									Parità sul carattere
I carattere	0	0	1	1	0	0	1	1	0
II carattere	1	0	1	1	1	0	1	0	1
III carattere	0	1	1	0	0	1	0	1	0
IV carattere	0	1	0	1	1	1	1	0	1
V carattere	1	1	0	1	0	1	0	1	1
LRC	0	1	1	0	0	1	1	1	1

Figura 1.13: Parità longitudinale

Si supponga che detto byte aggiuntivo sia formato dal XOR dei caratteri precedenti, ossia (in linguaggio C):

```
LRC = Testo[0]^Testo[1]^Testo[2]^Testo[3]^Testo[4];
```

Tale procedimento equivale a calcolare la parità pari di ciascuna colonna di bit del testo.

Il ricevente esegue lo stesso calcolo e confronta il risultato con la parità longitudinale ricevuta: se i due byte concordano è probabile che non siano stati trasmessi errori (oppure che ne siano stati introdotti in numero pari), se, invece, si evidenzia una discordanza fra LRC calcolato e quello ricevuto, si è sicuramente in presenza di un errore in fase di trasmissione.

Un tale sistema introduce gli stessi difetti del controllo di parità: errori pari sulla singola colonna non possono essere rilevati. Se però il controllo

di parità longitudinale è abbinato alla parità sul carattere, le cose migliorano sensibilmente.

									Parità sul carattere
I carattere	0	0	1	1	0	0	1	1	0
II carattere	1	~~0~~ 1	1	1	~~1~~ 0	0	1	0	1
III carattere	0	1	1	0	0	1	0	1	0
IV carattere	0	~~1~~ 0	0	1	~~1~~ 0	1	1	0	1
V carattere	1	1	0	1	0	1	0	1	1
LRC	0	1	1	0	0	1	1	1	1

Figura 1.14: Errori pari e parità longitudinale

Un numero pari di errori sulla stessa colonna produce lo stesso numero di errori di parità sul carattere, a meno che le colonne affette da errore pari non siano anch'esse in numero pari. Se si dovesse verificare una situazione simile a quella illustrata in fig. 1.14, non sarebbe possibile per il sistema rilevare alcun errore.

Naturalmente si devono verificare contemporaneamente una serie di sfortunate coincidenze, ma quando, nell'arco di mesi o anni, si trasmettono milioni o miliardi di caratteri, la "sfortunata coincidenza" è assolutamente inevitabile.

Ben diversa sarebbe la situazione rappresentata dalla fig. 1.15 e dalla fig 1.16, in cui gli errori assumono una distribuzione righe-colonne diversa e più accidentale.

Figura 1.15: Errori rilevati dalla parità longitudinale

In fig. 1.15 gli errori sono rilevati dalla parità longitudinale, perché gli errori, pur pari, non sono "allineati" in modo funzionale verticalmente al fine di mascherare gli errori.

In fig. 1.16 gli errori sono rilevati dalla parità sul carattere, perché, stavolta, gli errori non sono allineati orizzontalmente.

									Parità sul carattere
I carattere	0	0	1	1	0	0	~~1~~ 0	1	0
II carattere	1	~~0~~ 1	1	1	1	0	1	0	1
III carattere	0	1	1	0	0	1	~~0~~ 1	1	0
IV carattere	0	~~1~~ 0	0	1	1	1	1	0	1
V carattere	1	1	0	1	0	1	0	1	1
LRC	0	1	1	0	0	1	1	1	1

Figura 1.16: Errori rilevati dalla parità sul carattere

Assume quindi un certo interesse la situazione rappresentata in fig. 1.14 nella pagina precedente, al fine di valutare la probabilità che una simile situazione si avveri.

Due doppi errori sul carattere e due doppi errori sulla parità longitudinale rappresentano il più semplice dei casi in cui il doppio filtro delle due parità viene eluso.

Qualsiasi altra situazione non rilevata nè dalla parità sul carattere nè dalla parità longitudinale hanno una probabilità di molto minore di avverarsi, per cui si ritiene utile semplicare il problema analizzando solamente la più semplice delle situazioni d'errore.

La probabilità che si abbiano k errori pari sul singolo carattere si è visto essere espressa dalla seguente relazione

$$p_c = \frac{b!}{k!(b-k)!} p^k \tag{1.6}$$

dove p rappresenta la probabilità di errore sul singolo bit; b il numero di bit di ciascun carattere, esclusi il bit di start e quello di stop; k il numero di bit errati sul singolo carattere.

Affinché si possa realizzare la situazione di fig. 1.14 nella pagina precedente è necessario che un secondo carattere presenti due errori *negli stessi identici bit*. La prima condizione (due errori nello stesso carattere) è verificata dalla relazione 1.6, che esprime $\binom{b}{n}$ diversi modi di manifestarsi dell'errore.

La seconda condizione esprime, però, la necessità che i bit errati siano gli stessi del carattere precedente, in modo tale che gli errori siano allineati anche verticalmente.

Quindi uno solo dei diversi $\binom{b}{k}$ modi di diporsi degli errori va preso in considerazione, per cui, nel caso del secondo carattere la 1.6 va divisa per $\binom{b}{k}$, ottenendo in tal modo la seguente relazione finale, che tiene conto di tutti e

quattro gli errori:

$$p_t = \frac{b!}{k!(b-k)!}p^k \cdot p^k = \frac{b!}{k!(b-k)!}p^{2k} \tag{1.7}$$

Nel caso specifico, ossia ipotizzando che k valga 2, si ha:

$$p_t = \frac{9!}{2!(9-2)!}10^{-16} = \frac{362880}{10080}10^{-16} = 36 \cdot 10^{-16} \tag{1.8}$$

In termini probabilistici si è autorizzati ad ipotizzare un flusso di $2.8 \cdot 10^{14}$ caratteri, ovvero 28000 miliardi di caratteri, prima di dover considerare realistico l'avverarsi di una serie di errori che riescano ad eludere il doppio filtro della parità sul carattere e della parità longitudinale.

Si rammenta, a tal proposito, che l'uso della sola parità sul carattere prevedeva un errore ogni 2.8 milioni di caratteri. Quindi l'uso combinato delle due parità ha notevolmente migliorato il tasso d'errore della comunicazione.

1.5.4 Il codice Hamming

Si è visto che mediante il controllo di parità si possono rilevare gli errori in numero pari. Esistono anche dei codici in grado di rilevare un certo numero k di errori sui b bit del carattere: aumentando k aumenta naturalmente la ridondanza del codice.

In una larga classe di questi codici si cerca di rendere massimo un parametro noto come distanza di Hamming: definito come il numero di posizioni in cui i possibili caratteri differiscono tra di loro o, in altri termini, il numero degli errori necessari per cambiare un carattere in un altro. Ciò implica che non tutte le possibili combinazioni di un certo numero di bit possono essere utilizzate.

Quando vi è un errore singolo la distanza tra la configurazione ricevuta e quella trasmessa è 1. Se la distanza di Hamming del particolare codice usato è 3, può essere rilevato sia l'errore singolo che quello doppio, in quanto neppure la variazione di 2 bit porta ad un altro carattere significativo. In generale un codice con distanza di Hamming pari a k può rilevare $k-1$ errori nello stesso carattere.

La considerazione su cui si basa la costruzione dei codici che permettono la correzione degli errori è che il carattere risultante è più vicino ad uno particolare, di quelli ammessi, che a tutti gli altri.

Questa considerazione porta a concludere che utilizzando un codice con distanza 3, possono essere corretti solo gli errori singoli, in quanto tali caratteri avranno una distanza pari a 1 dal carattere di partenza e almeno 2 da tutti gli altri.

Si consideri, ad esempio, la tabella 1.2 nella pagina seguente. Essa contiene un codice formato da 16 caratteri da 7 bit. Ciascuno di essi è costituito da 4 bit di dato e 3 bit di controllo. In tal modo ciascun carattere differisce dagli altri per almeno 3 bit.

Se ogni singolo carattere è formato dai bit $C_2C_1C_0D_3D_2D_1D_0$, il calcolo dei bit di controllo avviene secondo le seguenti regole:

$$C_2 = D_3 + D_2 + D_1$$
$$C_1 = D_2 + D_1 + D_0$$
$$C_0 = D_3 + D_2 + D_0$$

La ridondanza di tale codice è piuttosto elevata e non sempre tollerabile, anche se diventa via via più accettabile man mano che aumenta la lunghezza del codice.

Controllo	Dato
000	0000
011	0001
110	0010
101	0011
111	0100
100	0101
001	0110
010	0111
101	1000
110	1001
011	1010
000	1011
010	1100
001	1101
100	1110
111	1111

Tabella 1.2: Codifica del dato mediante codice Hamming

Il codice Hamming potrebbe essere utilizzato con profitto in quelle applicazioni industriali dove si devono trasmettere in rete pochi comandi corredati da altrettanto pochi parametri. In tal caso questa soluzione, abbinata alle due parità analizzate nelle precedenti sezioni forniscono un alto grado di protezione.

Basti considerare che il doppio errore può essere eliminato, per cui possono eludere il filtro solamente quelle situazioni ove si presentano 4 errori sul singolo carattere.

In tal caso la probabilità di errore sale a:

$$p_4 = \frac{9!}{4!(9-4)!}10^{-32} = \frac{362880}{2880}10^{-32} = 1.26 \cdot 10^{-30} \tag{1.9}$$

che costituisce una probabilità di errore di tutta tranquillità in situazione di scarso traffico di rete.

1.5.5 I codici di ridondanza ciclici

Il sistema di controllo degli errori più usato utilizza i codici ciclici o polinomiali (CRC), che si dimostrano estremamente efficienti, riescono cioè a rilevare una maggiore percentuale di errori a fronte di una minore ridondanza.

Tale codice può essere lungo 16, 24 oppure 32 bit o oltre, ma solitamente 16 bit sono sufficienti per la maggior parte delle applicazioni.

Per la costruzione del CRC (*Cyclic Redundancy Checking*) viene normalmente utilizzata un'aritmetica in modulo 2 o, se si preferisce, in complemento a 2.

Il messaggio, lungo n, viene considerato come un polinomio e gli n bit, 0 o 1, sono i coefficienti di un polinomio di grado massimo $n-1$ nella variabile fittizia x.

La successione di bit 1101001, ad esempio, corrisponde al polinomio

$$1 \cdot x_6 + 1 \cdot x_5 + 0 \cdot x_4 + 1 \cdot x_3 + 0 \cdot x_2 + 0 \cdot x_1 + 1 \cdot x_0 = x_6 + x_5 + x_3 + 1$$

Per generare il CRC viene utilizzato anche un secondo polinomio detto generatore $\big(G(x)\big)$, che determina le caratteristiche del CRC stesso.

Il polinomio generatore deve avere grado inferiore a quello ottenuto nel modo visto e deve terminare con $1 \cdot x_0 = 1$.

Alcuni di questi polinomi generatori sono ormai considerati standard (come ad esempio $x_{16} + x_{12} + x_5 + 1$) per i protocolli HDLC e SDLC.

Il CRC viene calcolato moltiplicando il polinomio messaggio per x^k, essendo k il grado del polinomio generatore, in modo da azzerare le sue ultime k posizioni.

Il risultato viene diviso per il polinomio generatore ottenendo un determinato quoziente ed un resto, che rappresenta proprio il CRC. Questo viene sommato al polinomio risultato dalla moltiplicazione e la somma così ottenuta costituisce l'insieme dei bit da trasmettere.

In questo modo il polinomio risultante, se privato dell'ultima parte, ossia del CRC, è identico al polinomio messaggio, non essendo l'informazione stata modificata. Il ricevente non dovrà fare altro che dividere il polinomio ricevuto, compreso di CRC, per lo stesso polinomio generatore: il risultato dovrà essere con resto zero.

Vengono rilevati:

- tutti gli errori singoli;
- tutti gli errori doppi se $G(x)$ possiede almeno tre uni;
- qualsiasi numero dispari di errori, se $G(x)$ contiene il fattore $(x+1)$;
- tutti gli errori a burst che si estendono per k o meno bit.

Rimangono non rilevati solo configurazioni di errori che per caso danno una stringa di bit esattamente divisibile per il polinomio generatore.

I controlli ciclici sono facilmente realizzabili in hardware. Le operazioni di divisione e moltiplicazione in modulo 2 dei polinomi, vengono eseguite mediante degli shift register, shiftando rispettivamente a sinistra o a destra il polinomio, mentre la sottrazione e la somma, dato che tali operazioni vengono effettuate senza prestito né riporto, vengono eseguite mediante delle porte XOR.

La diffusione dei controlli ciclici ha permesso la nascita di polinomi generatori considerati standard, i più importanti dei quali sono indicati nella sottostante tabella.

Polinomio	Commento
$x^{16}+x^{15}+x^2+1$	CRC 16
$x^{16}+x^{15}+x+1$	CRC 16 inverso
$x^{12}+x^{11}+x^3+x^2+x+1$	CRC 12
$x^8+x^7+x^5+x^4+x+1$	CRC 8
x^8+1	LRC 8
$x^{16}+x^{12}+x^5+1$	CRC CCITT
$x^{16}+x^{11}+x^4+1$	CRC CCITT inverso

Tabella 1.3: Polinomi generatori maggiormente usati

Fra questi, particolare importanza rivestono i codici CRC16 e CRC CCITT.

1.5.5.1 Un esempio di CRC

Si supponga di dover trasmettere il seguente testo[5]:

0100 1001 0101 0100 0101 0011

Si supponga, inoltre, che si voglia proteggere detto testo mediante un controllo polinomiale utilizzando come polinomio generatore il polinomio

$$G(x) = x^{16} + x^{15} + x^7 + 1 \tag{1.10}$$

Il primo passo da fare è trasformare la sequenza di bit relativa al testo da trasmettere in polinomio:

$$0x^{23} + 1x^{22} + 0x^{21} + ... + 0x^2 + 1x^1 + 1x^0 \tag{1.11}$$

Il suddetto polinomio va moltiplicato per il grado del polinomio generatore. Rappresentando la sequenza di bit non in forma polinomiale, per ragioni di spazio, si ottiene:

0100 1001 0101 0100 0101 0011 0000 0000 0000 0000 (1.12)

dove la parte in grassetto rappresenta il testo originale.

La sequenza di bit rappresentata in 1.12 va divisa per il polinomio generatore, ottenendo il seguente quoziente

$$q = 0011\ 0000\ 1101\ 0010\ 0111\ 1011 \tag{1.13}$$

(equivalente al valore esadecimale 0x30D27B) ed il seguente resto

$$r = 0111\ 0000\ 0000\ 0101 \tag{1.14}$$

[5]La sequenza binaria equivale all'esadecimale 0x495453 che, secondo la tabella ASCII corrisponde alla stringa ITS.

(equivalente al valore esadecimale 0x7005).

Il resto così ottenuto va sommato alla sequenza 1.12, ottenendo

$$0100\ 1001\ 0101\ 0100\ 0101\ 0011\ 0111\ 0000\ 0000\ 0101 \tag{1.15}$$

che rappresenta la somma fra dividendo e resto ovvero la stringa che viene effettivamente trasmessa.

Chi riceve la stringa di bit divide la sequenza per lo stesso polinomio generatore e detta divisione, se la trasmissione è andata a buon fine, non deve produrre resto. In caso contrario significa che durante la trasmissione si è prodotto un errore.

Dopo aver eseguito la divisione e aver verificato che il resro vale zero, è sufficiente ridividere la sequenza per il grado del polinomio per riottenere il testo orginale.

Spiegato così sembra molto laborioso, ma in realtà è molto semplice e molto efficiente. Dopo aver verificato il CRC è sufficiente togliere dalla sequenza 1.15 i 16 bit meno significativi per ottenere il testo originale senza particolari sforzi.

Nella prossima sezione verrà brevemente illustrato un possibile algoritmo di calcolo del CRC 16, usato comunemente nelle comunicazione nel mondo Internet.

1.5.5.2 Un semplice algoritmo di calcolo del CRC

Uno dei polinomi più usati nell'ambito delle reti industriali è il CRC16, dato da

$$G(x) = x^{16} + x^{15} + x^2 + 1 \tag{1.16}$$

L'algoritmo di calcolo del CRC16 prevede i seguenti passi:

1. si inizializza il CRC al valore 0xFFFF;
2. si esegue l'XOR del byte di testo con il CRC e si pone il risultato nel CRC;
3. si esegue lo shift a destra di un bit del CRC e se ne azzera il MSB;
4. se il bit meno significativo del CRC prima dello shift era 0 si torna al punto 3 altrimenti si esegue l'XOR del CRC con il valore 0xA001;
5. i punti 3 e 4 devono essere ripetuti 8 volte per ogni byte;
6. si torna al punto 2 finché tutti i byte del testo sono stati processati;
7. il valore del CRC è il risultato

```
WORD CalcCRC16(char *frame, char framelgt)
{
   WORD temp;
   bool odd;
   WORD crc;
   int i,j;

   // 1. Si inizializza il CRC al valore 0xFFFF
   crc = 0xFFFF;
```

```
    // 6. Si ripete dal punto 2 fino a quanto tutti i byte del
    //    frame sono stati processati.
    for (i=0; i<framelgt; i++)
    {
       // 2. Si esegue lo XOR (OR esclusivo) del byte con il byte
       //    basso del CRC memorizzando il risultato nel CRC.
       temp = frame[i]&0x00FF;
       crc ^= temp;

       // 5. I punti 3 e 4 vanno ripetuti 8 volte per ogni byte
       for (j=0; j<8; j++)
       {
          // 3. Si esegue lo shift del CRC di una posizione verso
          //    destra e si pone a 0 il bit15 (MSB) del CRC.
          odd = crc&0x0001?true:false;
          crc = crc >> 1;

          // 4. Se il bit0 prima dello shift era 0 si torna al
          //    punto 3 mentre se il bit0 prima dello shift era
          //    1 si esegue l'XOR del CRC con il valore 0xA001.
          if (odd)
             crc ^= 0xA001;
       }
    }

    return crc;
}
```

La prima cosa che colpisce è l'assenza di moltiplicazioni e divisioni. Un motivo sufficientemente robusto per giustificare l'assenza di dette operazioni è dato dal fatto che il testo da proteggere potrebbe essere formato da centinaia o migliaia di MByte. Ciò significherebbe gestire polinomi con esponenti improponibili.

Inoltre la divisione utilizza efficacemente l'aritmetica in modulo 2, che utilizza l'operazione di XOR. In particolare, il resto della divisione del polinomio $B(x)$ per il polinomio $C(x)$ si ottiene eseguendo l'XOR su ciascuna coppia di coefficienti corrispondenti. Tale proprietà dell'aritmetica modulo 2 deriva dal fatto che i coefficienti possono assumere solamente i valori 0 e 1 e permette l'implementazione del sottostante circuito di calcolo del CRC16.
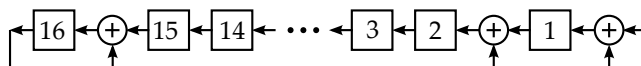
Figura 1.17: CRC16 hardware

Esso avviene attraverso degli shift a sinistra che alimentano tre XOR nei livelli 0, 2 e 15, che rappresentano il polinomio generatore. Quando tutti i bit del testo avranno attraversato lo shift, il contenuto dei 16 *flip-flop* rappresenterà il CRC.

Nell'algoritmo gli shift sono però effettuati verso destra ed il polinomio generatore appare invertito orizzontalmente (0xA001 anziché 0x8005). Ciò è dovuto al fatto che gli USART trasmettonon prima il bit meno significativo e

per ultimo quello più significativo, mentre in fig. 1.17 lo shift appare secondo logica matematica.

La qualità offerta dal CRC16 è piuttosto buona. Dato un testo di lunghezza $n \geq 32767$, ill CRC16 è in grado di rilevare:

- tutti gli errori singoli;
- tutti gli errori doppi;
- tutti gli errori tripli;
- tutti gli errori dispari;
- tutti gli errori a burst che si estendono per 16 bit o meno;
- il 99.997% degli errori a burst che si estendono per 17 bit;
- il 99.998% degli errori a burst che si estendono per 18 o più bit.

Il fatto che rilevi tutti gli errori dispari ha reso obsoleto l'uso della parità sul carattere, al punto che molti USART non la supportano più.

1.5.5.3 Un semplice algoritmo di calcolo del Checksum

Un ulteriore algoritmo molto usato, specialmente nel mondo Internet è il Checksum. Fornisce buone garanzie di affidabilità, anche se inferiori al CRC polinomiale. E' altrettanto semplice da implementare, basandosi su una somma ripetuta senza riporto.

Ciascun carattere, privato del bit di start, stop e parità, viene sommato, senza riporto e senza segno, mantenendo una larghezza di parola di 16 bit. Se, dopo l'ennesima somma, vi è riporto sul 17esimo bit, il risultato, troncato a 16 bit, viene semplicemente incrementato.

Il codice riportato di seguito, è un esempio di tale algoritmo.

```
u_short Checksum(u_short *buf, int count)
{
   u_long sum = 0;

   while(count--)
   {
      sum += *buf++;
      if(sum & 0xFFFF0000)
      {
         sum &= 0xFFFF;
         sum++;
      }
   }
   return ~(sum & 0xFFFF);
}
```

1.6 Esercizi

Gli esercizi riportati nelle seguenti pagine sono tutti relativi a quanto esposto nel capitolo 1.

1. ◊◊◊ Che differenza c'è fra un errore sintattico e un errore semantico?
2. ◊◊◊ Qual è il fine della ridondanza nelle comunicazioni?
3. ◊◊◊ Che differenza c'è fra una comunicazione parallela e una comunicazione seriale?
4. ◊◊◊ Qual è il principale vantaggio della comunicazione seriale rispetto alla parallela?
5. ◊◊◊ Cosa si intende con l'allocuzione inglese *baud rate*?
6. ◊◊◊ Come potrebbe essere tradotta in italiano la suddetta allocuzione?
7. ◊◊◊ Che differenza c'è fra comunicazione seriale sincrona e comunicazione seriale asincrona?
8. ◊◊◊ Qual è il principale vantaggio della comunicazione seriale sincrona su corte distanze (<30cm) rispetto a quella ascincrona?
9. ◊◊◊ Quali sono i due maggiori svantaggi della comunicazione seriale sincrona su lunghe distanze (>100cm) rispetto a quella asincrona?
10. ◊◊◊ Come avviene, genericamente, la sincronizzazione fra trasmettitore e ricevitore durante una comunicazione seriale asincrona?
11. ◊◊◊ Da cosa è caratterizzata una trasmissione sbilanciata?
12. ◊◊◊ In quali casi è indicata?
13. ◊◊◊ Quali sono le principali caratteristiche di una trasmissione bilanciata?
14. ◊◊◊ In quali casi è indicata?
15. ◊◊◊ Si codifichi l'informazione relativa all'esercizio precedente in codifica Manchester.
16. ◊◊◊ Si evidenzino i principali pregi e difetti della codifica Mancester rispetto alla NRZ.
17. ◊◊◊ Si supponga che, incorniciati fra i bit di Start e di Stop, vengano trasmessi i bit 0 1 1 0 1 1 1 0 0. Si tratta di una trasmissione in parità pari o in parità dispari?
18. ◊◊◊ A che numero decimale corrisponde il polinomio $x^{16} + x^{15} + x^2 + 1$?
19. ◊◊◊ Il numero dell'esercizio precedente è un numero primo?

20. ◊◊♦ Si suppongano due orologi e un collegamento elettrico, come illustrato nella figura 1.18.

    Da A viene spedita un'informazione di 8 bit (0xAA) verso B, senza l'uso né di bit di parità né di stop. La sincronizzazione dei due orologi viene effettuata sul fronte di discesa del bit di sincronizzazione (attivo basso) e il periodo di bit è di un secondo. Si suppone l'orologio posto in A di precisione ideale. Qual è l'errore massimo ammissibile al fine di evitare errori di comunicazione da parte dell'orologio B espresso in secondi al giorno?
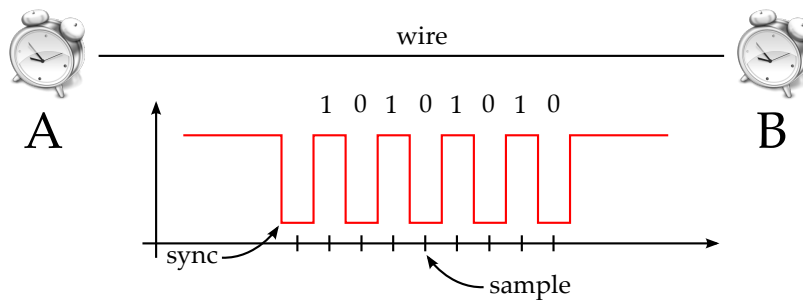
Figura 1.18: Comunicazione asincrona

Bibliografia

[1] Denver Allen, *Serial Communications*, Microsoft Windows Developer Support, 1995